The Minimal Cost Reachability Problem in Priced Timed Pushdown Systems

Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman

Uppsala University, Sweden

Abstract. This paper introduces the model of *priced timed pushdown systems* as an extension of discrete-timed pushdown systems with a cost model that assigns multidimensional costs to both transitions and stack symbols. For this model, we consider the minimal cost reachability problem: i.e., given a priced timed pushdown system and a target set of configurations, determine the minimal possible cost of any run from the initial to a target configuration. We solve the problem by reducing it to the reachability problem in standard pushdown systems.

1 Introduction

Pushdown systems are one of the most widely used models for the study and analysis of recursive systems [13]. Furthermore, several models have been introduced in [8, 11, 7, 12, 18] which extend the model by introducing timed behaviors.

We consider a new model for *Timed Pushdown Systems* consisting of pushdown systems augmented with a finite set of (integer valued) clocks. Moreover, the symbols that are pushed into the stack have integer ages measuring the time that has elapsed since they were pushed. A clock can be set to zero simultaneously with any transition. At any moment, the value of a clock is the time elapsed since the last time it was reset. With each transition we associate time-intervals (whose bounds are natural numbers or ω) that restrict the clock values and ages of the sequence of symbols (on the top of the stack) that can be popped.

In parallel, there have been several works on extending the model of timed automata [4] with prices (weights) (see e.g., [10, 5]). Priced timed automata are suitable models for embedded systems, where we have to take into consideration the fact that the system behavior may be constrained by the consumption of different types of resources. More precisely, priced timed automata extend classical timed automata with a cost function *Cost* that maps every location and every transition to a nonnegative integer. For a transition, *Cost* gives the cost of performing the transition. For a location, *Cost* gives the cost per time unit for staying in the location. In this manner, we can define, for each run of the system, the accumulated cost of staying in locations and performing transitions.

We study a natural extension of timed pushdown systems, namely *Priced Timed Pushdown Systems* (PTPS). We allow the cost function to map transitions and stack symbols of the pushdown system into vectors of integers (of some given length k). Again, for a transition, *Cost* gives the cost of performing the transition;

while for a stack symbol, *Cost* gives the cost per time unit for the symbol to stay in the stack. We consider the *minimal cost reachability problem* for PTPS where, given an *intial configuration* c_0 and a set F of *final configurations*, the task is to compute the minimal accumulated cost of a run that reaches F from c_0. Here, we assume that F is *regular* (i.e., F can be described using a finite state automaton). Since the set of costs within which we can reach F from c_0 is upward closed (regardless of the form of F), the minimal cost reachability problem can be reduced, using the construction of Valk and Jantzen [19], to the *cost-threshold problem*. In the latter, we are given a cost vector $\mathbf{v}$, and we want to check if it is possible to reach F from c_0 with a cost that does not exceed $\mathbf{v}$.

In this paper, we prove that the cost-threshold problem for PTPS can be reduced to the reachability problem for (unpriced) timed pushdown systems. The idea consists of encoding the cost of a computation in the state of the timed pushdown systems. Moreover, we show that the reachability problem for timed pushdown systems can be reduced to the reachability for (unpriced and untimed) pushdown systems which is decidable. Hence, we get the decidability of the cost-threshold problem for PTPS (and consequently also solving the minimal cost reachability problem for PTPS).

Related work. The works in [8, 11, 12] consider timed pushdown automata. However, the models in these works consider only global clocks which means that the stack symbols are not equipped with clocks. In contrast, we associate with each pushed stack symbol one clock (reflecting its age). In fact, our model can be easily extended such that each stack symbol has several clocks.

In [18], the authors introduce *recursive timed automata*, a model where clocks are considered as variables. A recursive timed automaton allows passing the values of clocks using either *pass-by-value* or *pass-by-reference* mechanisms. This feature is not supported in our model since we do not allow pass-by-value communication between procedures. Moreover, in the recursive timed automaton model, the local clocks of the caller procedure are stopped until the called procedure returns. This makes the semantics of the models incomparable with ours, since all the clocks in our model evolve synchronously.

In [7], the authors define the class of *extended pushdown timed automata* which is a pushdown automaton enriched with a set of clocks, with an additional stack used to store/restore clock valuations (which leads to the undecidability of the reachability problem). In our model, clocks are associated with stack symbols and store/restore operations are not allowed.

None of the above works considers prices in their models.

The minimal cost reachability problem has been addressed for several models: priced timed automata (e.g., [10, 5]), and priced timed Petri nets ([2, 1]). To the best of our knowledge, this is the first work that addresses the problem for PTPS.

2 Preliminaries

Let $\mathbb{N}$ denote the non-negative integers, and let $\mathbb{N}^k$ and $\mathbb{N}_\omega^k$ denote the set of vectors of dimension k over $\mathbb{N}$ and $\mathbb{N} \cup \{\omega\}$, respectively ($\omega$ represents the first

limit ordinal). We use $\mathbf{0}^k$ (or simply $\mathbf{0}$, depending on the context) to denote the vector of dimension k whose elements have all the value 0, and *Intrv* to denote the set of intervals in $\mathbb{N} \times \mathbb{N}_\omega$. In the context of vectors, *less than* means the standard componentwise ordering $\leq$.

For sets A and B, we use $f : A \to B$ to denote that f is a total function that maps A to B. We use $[A \to B]$ to denote the set of all total functions from A to B. Given a set A with an ordering $\preceq$ and a subset $B \subseteq A$, B is said to be *upward closed* in A if $b \in B$, $a \in A$ and $b \preceq a$ implies $a \in B$. Given a set $B \subseteq A$, we define the *upward closure* $B\uparrow$ (resp. *downward closure* $B\downarrow$) to be the set $\{a \in A \mid \exists b \in B : b \preceq a \, (resp.\, a \preceq b)\}$.

Let Σ be an alphabet. We denote by Σ^* (resp. Σ^+) the set of all *words* (resp. non-empty words) over Σ, and by ϵ the empty word. A language is a (possibly infinite) set of words. The length of a word $w \in \Sigma^*$ is denoted by $|w|$. (We assume that $|\epsilon| = 0$). For every $i : 1 \leq i \leq |w|$, let $w(i)$ be the symbol at position i in w. For $a \in \Sigma$, we write $a \in w$ if a appears in w, i.e., $a = w(i)$ for some $i : 1 \leq i \leq |w|$. We use $|w|_a$ to denote the number of occurrences of a in w.

3 Priced Timed Pushdown Systems

The *Timed Pushdown System* (TPS) model is an extension of pushdown systems augmented with a finite set of (integer valued) clocks. Moreover, the symbols that are pushed into the stack have integer ages measuring the time that has elapsed since they were pushed. A clock can be set to zero simultaneously with any transition. At any moment, the value of a clock is the time elapsed since the last time it was reset. With each transition we associate time-intervals (whose bounds are natural numbers or ω) which restrict the clock values and ages of the sequence of symbols (on the top of the stack) that can be popped. Then, we extend this model to priced timed pushdown systems (PTPS) by assigning multidimensional costs to both transitions (action costs) and stack symbols (storage costs). Each firing of a discrete transition costs the assigned cost vector. The cost of a timed transition depends on the stack content. For each stack symbol γ, if the stack contains k_1 occurrences of γ and the cost of storing γ per time unit is $\mathbf{v}$, then firing a timed transition will add $k_1\mathbf{v}$ to the current accumulated cost.

Syntax. A Priced Timed Pushdown System (PTPS) is a tuple $N = (X, Q, \Gamma, \Delta, cost)$ where X is a finite set of clocks, Q is a finite set of states, Γ is the stack alphabet, $cost : (\Gamma \cup \Delta) \to \mathbb{N}^k$ is a function assigning (multidimensional) firing costs to transitions and storage costs to stack symbols, and Δ is a finite set of transition rules of the form:

$$(q, (\alpha_1, J_1) \cdots (\alpha_m, J_m)) \xrightarrow{(\phi\,,\, R)} (q', (\gamma_1, I_1) \cdots (\gamma_n, I_n))$$

where (1) $q, q' \in Q$ are two states, (2) $\alpha_1 \cdots \alpha_m \in \Gamma^*$ is the word to be popped such that the age of each symbol $\alpha_i \in \Gamma$ should be in the interval J_i for all $i : 1 \leq i \leq m$, (3) $\phi : X \to Intrv$ is a clock constraint over X (i.e., the valuation of a clock $x \in X$ should be in $\phi(x)$), (4) $R \subseteq X$ is the set of clocks to be reset,

and (5) $\gamma_1 \cdots \gamma_n \in \Gamma^*$ is the word to be pushed into the stack such that the initial age of each symbol $\gamma_j \in \Gamma$ should be in the interval I_j for all $j : 1 \leq j \leq n$. Observe that the set Δ of transition rules can be defined as a finite subset of $\big(Q \times (\Gamma \times \mathit{Intrv})^*\big) \times \big([X \to \mathit{Intrv}] \times 2^X\big) \times \big(Q \times (\Gamma \times \mathit{Intrv})^*\big)$.

If for every $\gamma \in \Gamma$ and $\delta \in \Delta$, $\mathit{cost}(\gamma) = \mathit{cost}(\delta) = \mathbf{0}$ (i.e., the cost of any transition or stack symbol is $\mathbf{0}$), then N is called a *(unpriced) Timed Pushdown System* (TPS), which can be described by the tuple (X, Q, Γ, Δ). Moreover, if $\Delta \subseteq \big(Q \times (\Gamma \times I_0)^*\big) \times ([X \to I_0] \times 2^X) \times (Q \times (\Gamma \times I_0)^*)$ with $I_0 = \{[0..\omega]\}$, then the TPS N is called an *unpriced and untimed pushdown system*, or simply Pushdown System (PS), which can be described by the tuple (Q, Γ, Δ).

Configurations. A configuration c of N is a triple (q, ν, w) where $q \in Q$ is a state, $\nu : X \to \mathbb{N}$ is a clock valuation, and $w \in (\Gamma \times \mathbb{N})^*$ is the stack content. Observe that the stack contains a sequence of pairs representing the pushed symbols and their ages. Let $\mathit{Conf}(N)$ denote the set of all configurations of N.

A set of configurations $C \subseteq \mathit{Conf}(N)$ is said to be *regular* if there are a set $Q' \subseteq Q$ of states, a **finite** set $\Psi \subseteq [X \to \mathbb{N}]$ of clock valuations, and a **regular** language L over $(\Gamma \times \mathit{Intrv})$ such that $C = \{(q, \nu, w) \,|\, q \in Q', \nu \in \Psi, w \text{ satisfies some } l \in L\}$. A language L over $(\Gamma \times \mathit{Intrv})$ is regular if and only if there is a finite state automaton $\mathcal{A}$ over the alphabet $(\Gamma \times \mathit{Intrv})$ such that the language accepted by $\mathcal{A}$ is precisely L.

Let $c = (q, \nu, (\gamma_1, a_1) \cdots (\gamma_n, a_n))$ be a configuration of N with $q \in Q$, $\nu \in [X \to \mathbb{N}]$, and $(\gamma_i, a_i) \in \Gamma \times \mathbb{N}$ for all $i : 1 \leq i \leq n$. Then, we use c^{+1} to denote the configuration (q, ν', w') defined as follows: (1) $\nu'(x) = \nu(x) + 1$ for all $x \in X$ (i.e., the value of each clock is increased by one time unit), and (2) $w' = (\gamma_1, a_1 + 1) \cdots (\gamma_n, a_n + 1)$ (i.e., the age of each symbol γ_i in the stack is also increased by one time unit). Note that (γ_1, a_1) is at the top and (γ_n, a_n) is at the bottom of the stack.

Let $\gamma_1, \ldots, \gamma_n \in \Gamma$, $a_1, \ldots, a_n \in \mathbb{N}$ and $I_1, \ldots, I_n \in \mathit{Intrv}$. We say that the stack content $w = (\gamma_1, a_1) \cdots (\gamma_n, a_n)$ satisfies the stack constraint $r = (\gamma_1, I_1) \cdots (\gamma_n, I_n)$ (denoted by $w \in r$) if and only if $a_i \in I_i$ for all $i : 1 \leq i \leq n$ (i.e., the age of each symbol γ_i belongs to the interval I_i).

Transition relation. We define two transition relations on the set of configurations of N: timed and discrete. The timed transition relation increases the value of each clock and the age of each pushed stack symbol by one. Formally, $c \to_{\mathit{time}} c'$ if and only if $c' = c^{+1}$.

We define the discrete transition relation $\to_D$ as $\bigcup_{\delta \in \Delta} \to_\delta$ where $\to_\delta$ represents the effect of *performing* the discrete transition δ. More precisely, let us assume that the transition δ is of the form $(q, r) \xrightarrow{(\phi, R)} (q', r')$ where $q, q' \in Q$ are states, $r, r' \in (\Gamma \times \mathit{Intrv})^*$ are stack constraints, $R \subseteq C$ is the set of clocks to be reset, and $\phi : X \to \mathit{Intrv}$ is a clock constraint over X. For configurations $c = (q, \nu, w)$ and $c' = (q', \nu', w')$, we have $c \to_\delta c'$ if and only if

- There are $u, u'' \in (\Gamma \times \mathbb{N})^*$ such that $w = u \cdot u''$ and $u \in r$. The transition t can be performed only if there is a word u, at the top of the stack, satisfying the constraint r (i.e., $u \in r$), and if this is the case u can be popped.

- There is $u' \in (\Gamma \times \mathbb{N})^*$ such that $w' = u' \cdot u''$ and $u' \in r'$. The newly pushed word u' into the stack should satisfy the stack constraint given by r'.
- $\nu(x) \in \phi(x)$ for all $x \in X$. The current clock value of x should satisfy the time constraint imposed by ϕ.
- $\nu'(x) = 0$ for all $x \in R$ and $\nu'(x') = \nu(x')$ for all $x' \notin R$. The clocks in R are reset to 0, and thus start counting time with respect to the time of occurrence of this transition.

We write $\rightarrow_N$ (or simply $\rightarrow$ when it is clear from the context) to denote the transition relation given by $\rightarrow_{time} \cup \rightarrow_D$. We use $\rightarrow^*$ to denote the reflexive-transitive closure of $\rightarrow$. It is easy to extend $\rightarrow^*$ to sets of configurations.

Let $c, c' \in \mathit{Conf}(N)$. A computation π of N from c to c' is of the form $c_0 \rightarrow c_1 \rightarrow \cdots \rightarrow c_n$ where $c_0 = c$, $c_n = c'$, and $c_i \rightarrow c_{i+1}$ for all $i : 0 \leq i < n$. We write $c \xrightarrow{\pi} c'$ to denote that there is a computation π of N from c to c'. We define $\mathit{Reach}(c) := \{c'' \mid c \rightarrow^* c''\}$ as the set of configurations reachable from c.

The cost of computations. The cost of a stack content $w \in (\Gamma \times \mathbb{N})^*$ is defined as $\mathit{Cost}(w) = \sum_{(\gamma,a) \in \Gamma \times \mathbb{N}} |w|_{(\gamma,a)} \; \mathit{cost}(\gamma)$. Intuitively, the cost of w corresponds to the sum, over the stack symbols $\gamma \in \Gamma$, of the number of occurrences of γ in w multiplied by its individual cost $\mathit{cost}(\gamma)$. In the same way, we can define the cost of stack constraints: Given a stack constraint $r \in (\Gamma \times \mathit{Intrv})^*$, $\mathit{Cost}(r) = \mathit{Cost}(u)$ for some $u \in r$. Notice that the function Cost over stack constraints is well-defined since $\mathit{Cost}(u) = \mathit{Cost}(u')$ for all $u, u' \in r$.

The cost of a discrete transition δ is defined as $\mathit{Cost}(c \rightarrow_\delta c') = \mathit{cost}(\delta)$ and the cost of a timed transition is defined as $\mathit{Cost}(c \rightarrow_{time} c^{+1}) = \mathit{Cost}(w)$ where c is of the form (q, ν, w). The cost of a computation $\pi = c_0 \rightarrow c_1 \rightarrow \cdots \rightarrow c_n$ is the sum of all transition costs, i.e., $\mathit{Cost}(\pi) = \sum_{i=0}^{n-1} \mathit{Cost}(c_i \rightarrow c_{i+1})$.

The Cost-Threshold Problem. We study the problem of computing the minimal cost for reaching a configuration in a given regular target set.

Cost-Threshold Problem

Instance: A PTPS $N = (X, Q, \Gamma, \Delta, \mathit{cost})$ with an initial configuration $c_0 \in \mathit{Conf}(N)$, a regular set F of final configurations and a vector $\mathbf{v} \in \mathbb{N}_\omega^k$.

Question: Does there exist $c \in F$ and $c_0 \xrightarrow{\pi} c$ such that $\mathit{Cost}(\pi) \leq \mathbf{v}$?

The cost-threshold problem is called the reachability problem when the PTPS N is an unpriced (un)timed pushdown system. In fact, in the case of unpriced (un)timed pushdown systems, the cost-threshold problem boils down to checking whether there exist a configuration $c \in F$ and a computation π such that $c_0 \xrightarrow{\pi} c$. Moreover, in the case of pushdown systems, the regular set F of final configurations can be defined using the tuple $Q' \times L$ where $Q' \subseteq Q$ and L is a regular language over Γ.

Since all costs are non-negative (in the set $\mathbb{N}^k$), the standard componentwise ordering $\leq$ on costs is a well-quasi order and thus every upward closed set of costs has finitely many minimal elements [14]. Moreover, if we have a positive instance of the cost-threshold problem with some allowed cost $\mathbf{v}$ then any modified instance with some allowed cost $\mathbf{v}' \geq \mathbf{v}$ will also be positive. Thus the set of

possible costs in the cost-threshold problem is upward-closed. In this case, the Valk-Jantzen theorem [19] implies that the set of minimal possible costs can be computed if the Cost-Threshold problem is decidable.

Theorem 1. *[Valk and Jantzen [19]] Given an upward-closed set $\mathbf{V} \subseteq \mathbb{N}^k$, the finite set $\mathbf{V}_{min}$ of minimal elements of $\mathbf{V}$ is computable if for any vector $\mathbf{v} \in \mathbb{N}^k_\omega$ the predicate $\mathbf{v}{\downarrow} \cap \mathbf{V} \neq \emptyset$ is decidable.*

Computing Minimal Possible Costs

Instance: A PTPS N with an initial configuration c_0 and a regular set F of final configurations.

Question: Compute the minimal possible costs of reaching F, i.e., the finitely many minimal elements of $\mathbf{V}_F = \{\mathbf{v} \in \mathbb{N}^k \mid \exists c \in F, \pi.\, c_0 \xrightarrow{\pi} c \wedge \mathit{Cost}(\pi) \leq \mathbf{v}\}$.

Since $\mathbf{V}_F$ is upward-closed and for any vector $\mathbf{v}$, if F is reachable with a cost less than $\mathbf{v}$, then $\mathbf{v} \in \mathbf{V}_F$, we have by Theorem 1:

Theorem 2. *Let N be a PTPS with an initial configuration c_0 and a regular set of final configurations F. Then, computing the minimal possible costs of reaching F can be reduced to the cost-threshold problem of reaching F from c_0.*

4 From Priced to Unpriced Timed Pushdown Systems

In this section, we show that it is possible to reduce the cost-threshold reachability problem for PTPS to the reachability problem for TPS.

Theorem 3. *The cost-threshold reachability problem for priced timed pushdown systems can be reduced to the reachability problem for timed pushdown systems.*

The rest of this section is devoted to the proof of this theorem. Consider an instance of the cost-threshold reachability problem for priced timed pushdown systems. Let $N = (X, Q, \Gamma, \Delta, \mathit{cost})$ be a PTPS with $\mathit{cost}(\Gamma) \cup \mathit{cost}(\Delta) \subseteq \mathbb{N}^k$, $c_0 \in \mathit{Conf}(N)$ be the initial configuration, F be the regular set of final configurations, and $\mathbf{v} = (v_1, \ldots, v_k) \in \mathbb{N}^k_\omega$.

First, for every $i : 1 \leq i \leq k$, if $v_i = \omega$ then we replace v_i by 0 and set the i-th component of the cost function *cost* of each stack symbol $\gamma \in \Gamma$ and transition $\delta \in \Delta$ to 0. Hence, we can assume that $\mathbf{v} = (v_1, \ldots, v_k) \in \mathbb{N}^k$.

In the following, we construct a TPS $N' = (X', Q', \Gamma', \Delta')$ such that the cost-threshold reachability for N is reducible to the reachability problem for N'. The idea is to simulate any computation of N by a computation of N'. During the simulation, N' keeps track in its state of the current total cost of the computation (performed so far) and the current cost of the stack content. So, when a discrete transition δ is performed, N' adds the cost of δ to the current total cost. (Observe that the total cost should always be less than the vector $\mathbf{v}$ since we are only interested in computations whose total cost is less than $\mathbf{v}$.) Now, when a timed transition is performed, the cost of the stack content (if it is less than $\mathbf{v}$) will be added to the total cost.

The first main difficulty is that, during one time unit, the cost of the stack content can be strictly greater than (or incomparable to) $\mathbf{v}$. To overcome this difficulty, N' keeps track of the cost of the stack content up to $\mathbf{v}$, and uses the special symbol $\top$ when the current cost of the stack content is not less than $\mathbf{v}$. Moreover, N' adds the cost of the current stack (stored in the state of N') to each newly pushed stack symbol $\gamma \in \Gamma$. Hence, a stack symbol of N' is of one of the following two forms: either $(\gamma, \mathbf{v}')$ or $(\gamma, \top)$ where $\gamma \in \Gamma$ and $\mathbf{v}' \leq \mathbf{v}$. Then, every transition performed by N' preserves the invariant between the stack cost stored in the current state of N' and the topmost stack symbol in the stack. This means that if the current cost of the stack content stored in the state is s and the topmost stack symbol is (γ, s') then the following condition should be satisfied: If $s' = \top$ or $cost(\gamma) + s' \not\leq \mathbf{v}$ then $s = \top$; otherwise $s = cost(\gamma) + s'$.

The second difficulty is that timed transitions of N' are performed in a non-deterministic manner. However, N' needs to know when a timed transition has been performed in order to add the current stack content cost to the total cost stored in its state. For this we add a new clock x_{new} which is used to detect if one unit of time has elapsed (i.e., a timed transition has been performed) or not. Then, discrete transitions of N can only be simulated by N' when the value of the clock x_{new} is equal to 0. If the current value of x_{new} is 1, then N' will add the current stack content cost (if it is less than $\mathbf{v}$) to the total cost stored in its state and reset x_{new}. Formally, N' is defined as follows:

- Let $\top$ be a symbol. The stack alphabet Γ' of N' is defined by the set $\Gamma \times (\mathbf{v}\downarrow \cup \{\top\})$. Intuitively, a stack symbol of the form $(\gamma, \mathbf{v}')$ (resp. $(\gamma, \top)$) corresponds to the fact that the cost of the stack content before pushing the stack symbol $(\gamma, \mathbf{v}')$ (resp. $(\gamma, \top)$) into the stack is $\mathbf{v}'$ (resp. not less than $\mathbf{v}$).
- A state of N' is of the form (q, t, s) where $q \in Q$ is a state of N, $t \in \mathbf{v}\downarrow$ is the current accumulated total cost (which should be less than $\mathbf{v}$), and $s \in (\mathbf{v}\downarrow \cup \{\top\})$ reflects the current cost of the stack content (if the current cost $\mathbf{v}'$ of the stack content is less than $\mathbf{v}$ then $s = \mathbf{v}'$ otherwise $s = \top$).
- The set of clocks of N' contains all the clocks of N and a new clock x_{new} such that $x_{new} \notin X$ (i.e., $X' = X \cup \{x_{new}\}$). The clock x_{new} is used to detect when a timed transition is performed in order to add the cost of the current stack content (if it is less than $\mathbf{v}$) to the total cost.
- The set Δ' is the smallest set of rules satisfying the following conditions:

  1. Simulating a discrete transition of N: Let $t, t' \in (\mathbf{v}\downarrow)$ be two vectors less than $\mathbf{v}$ and $s, s' \in (\mathbf{v}\downarrow \cup \{\top\})$. For every discrete transition $\delta \in \Delta$ of the form $(q, (\alpha_1, J_1) \cdots (\alpha_m, J_m)) \xrightarrow{(\phi, R)} (q', (\gamma_1, I_1) \cdots (\gamma_n, I_n))$, the TPS N' has a transition of the form $((q, t, s), r) \xrightarrow{(\phi', R)} ((q', t', s'), r')$, with $r = ((\alpha_1, a_1), J_1) \cdots ((\alpha_m, a_m), J_m)$ and $r' = ((\gamma_1, b_1), I_1) \cdots ((\gamma_n, b_n), I_n)$, if the following conditions are satisfied:

  - $t' := t + cost(\delta)$. The cost of the transition δ is add to the total cost t.
  - $\phi'(x) = \phi(x)$ if $x \in X$, and $\phi'(x_{new}) = [0..0]$. The discrete transition can be performed by N' only if the clock value of x_{new} is equal to 0.

- Let $d = s$ if $r = \epsilon$ (i.e., $m = 0$) otherwise $d = a_m$. Intuitively, d represents the stack cost after popping a word satisfying the stack constraint r.
  - If $d = \top$ then $s' = \top$ and $b_i = \top$ for all $i : 1 \leq i \leq n$. In fact, if the current cost of the stack after popping a word satisfying the constraint r is not less than $\mathbf{v}$ (since $d = \top$), then it is the case after pushing any stack symbol (so, $b_i = \top$ and $s' = \top$).
  - If $d \leq \mathbf{v}$ then for every $i : 1 \leq i \leq n$, let $\mathbf{v_i} = d + \sum_{j=i}^{n} cost(\gamma_j)$ be the sum of d (i.e, the current stack cost after popping a word satisfying the constraint r) and the cost of the sequence of stack symbols $\gamma_i\gamma_{i+1}\cdots\gamma_n$. Then, if $\mathbf{v_1} \leq \mathbf{v}$ then $s' = \mathbf{v_1}$, otherwise $s' = \top$. Moreover, $b_n = d$ and for every $i : 1 \leq i < n$, if $\mathbf{v_{i+1}} \leq \mathbf{v}$ then $b_i = \mathbf{v_{i+1}}$, otherwise $b_i = \top$.

2. Simulating a timed transition of N: For every state $q \in Q$ and vectors $t, t', s \in (\mathbf{v}\downarrow)$ less than $\mathbf{v}$, the TPS N' has a transition δ_{time} of the form $((q,t,s),\epsilon) \xrightarrow{(\phi,R)} ((q,t',s),\epsilon)$ if the following conditions hold:

- $\phi(x) = [0..\omega]$ for all $x \in X$ and $\phi(x_{new}) = [1..1]$. This means that one time unit has passed, and hence, a timed transition has been performed.
- $t' := t + s$. The current cost of the stack content is added to the current total cost since a timed transition has been performed.
- $R = \{x_{new}\}$. Only the clock x_{new} is reset to 0.

Relation between N and N'. Let $w = ((\gamma_1, a_1), y_1)((\gamma_2, a_2), y_2)\cdots((\gamma_n, a_n), y_n)$ be a possible stack content of N' where $\gamma_i \in \Gamma$, $a_i \in (\mathbf{v}\downarrow) \cup \{\top\}$, and $y_i \in \mathbb{N}$ for all $i : 1 \leq i \leq n$. Recall that the symbol $((\gamma_n, a_n), y_n)$ is in the bottom of the stack and $((\gamma_1, a_1), y_1)$ is the topmost stack symbol. For every $i : 1 \leq i \leq n$, let $\mathbf{v_i} = \sum_{j=i}^{n} cost(\gamma_j)$ be the cost of the sequence of stack symbols $\gamma_i \cdots \gamma_n$.

Then, w is a *valid* stack content of N' if and only if $a_n = \mathbf{0}$ and for every $i : 1 \leq i < n$, if $\mathbf{v_{i+1}} \leq \mathbf{v}$ then $a_i = \mathbf{v_{i+1}}$, otherwise $a_i = \top$. Observe that the transition relation of N' preserves the validity of the stack content in any configuration reachable from a configuration whose stack content is initially valid.

Now, we define the mapping T that associates, for every configuration $c = (q, \nu, (\gamma_1, y_1)\cdots(\gamma_n, y_n))$ of N and every vector $t \leq \mathbf{v}$, a configuration $T(c, t, \mathbf{v}) = ((q,t,s), \nu', w')$ of N' such that the following conditions are satisfied:

- w' is the unique valid stack content of the form:

$$((\gamma_1, a_1), y_1)((\gamma_2, a_2), y_2)\cdots((\gamma_n, a_n), y_n)$$

  Notice that such a valid stack configuration exists by definition and unique.
- If $Cost(\gamma_1 \cdots \gamma_n) \leq \mathbf{v}$ then $s = Cost(\gamma_1 \cdots \gamma_n)$, otherwise $s = \top$.
- $\nu'(x) = \nu(x)$ if $x \in X$ and $\nu'(x_{new}) = 0$.

Observe that T is a bijection. The definition of the mapping T is extended in the straightforward manner to sets of configurations of N and costs less than $\mathbf{v}$. Finally, Theorem 3 is an immediate consequence of the following lemma:

Lemma 4. *Let $F' = T(F, \mathbf{v}\downarrow, \mathbf{v})$. There exists a configuration $c \in F$ and a computation $c_0 \xrightarrow{\pi} c$ of N s.t. $Cost(\pi) \leq \mathbf{v}$ iff there is a configuration $c' \in F'$ s.t. $T(c_0, \mathbf{0}, \mathbf{v}) \rightarrow_{N'}^{*} c'$. Moreover, the set F' of configurations of N' is regular.*

5 From Timed Pushdown Systems to Pushdown Systems

In this section, we show that it is possible to reduce the reachability problem for TPS to its corresponding problem for PS. Let us first show that the reachability problem for TPS can be reduced to the reachability problem for TPS between two configurations with empty stack and where the value of each clock is 0.

Lemma 5. *Let N be a timed pushdown system, $c_0 \in Conf(N)$ be an initial configuration, and $F \subseteq Conf(N)$ be a regular set of final configurations. Then, it is possible to construct a timed pushdown system $N' = (X', Q', \Gamma', \Delta')$ and two configurations $c'_0 = (q_0, \nu, \epsilon)$ and $c'_f = (q_f, \nu, \epsilon)$ such that $q_0, q_f \in Q'$, $\nu(x) = 0$ for all $x \in X'$, and there is $c \in F$ such that $c_0 \rightarrow^*_N c$ iff $c'_0 \rightarrow^*_{N'} c'_f$.*

Proof. The proof of this lemma is similar to the case of standard pushdown systems. In fact, any computation of N' will be divided in three phases. In the first phase, the TPS N' performs some push and nop transitions in order to reach the configuration c_0. Then, N' starts to mimic the behavior of N. Finally, N' performs a sequence of pop and nop transitions to check, in a nondeterministic way, whether the current reached configuration is in F, and then resets all clocks to zero. This can be done since F is a regular set of configurations. □

Let us now prove that it is possible to reduce the reachability problem for timed pushdown systems to the reachability problem for pushdown systems.

Theorem 6. *The reachability problem for timed pushdown system can be reduced to the same problem for pushdown systems.*

The rest of this section is devoted to the proof of Theorem 6. Consider an instance of the reachability problem for timed pushdown systems: Let $N = (X, Q, \Gamma, \Delta)$ be a timed pushdown system, $c_0 \in Conf(N)$ be an initial configuration, F be a regular set of final configurations. From Lemma 5, we can assume without loss of generality that $c_0 = (q_0, \nu, \epsilon)$ and $F = \{(q_f, \nu, \epsilon)\}$ where $q_0, q_f \in Q$ and $\nu(x) = 0$ for all $x \in X$.

Let max be the maximal natural number appearing in the time intervals of the transition relation Δ. Observe that if the value of a clock or the age of a stack symbol is strictly greater than max then we can assume without loss of generality that it is ω.

In the following, we construct a pushdown system $N' = (Q', \Gamma', \Delta')$ such that the reachability problem for N is reducible to the reachability problem in N'. The main idea is to simulate a computation of N by a computation of N'. During the simulation process, the pushdown system keeps track of the value of each clock of N in its state up to the value max. In fact, if the value of a clock of N is strictly greater than max, N' can assume without loss of generality that the value of such clock is ω. Observe that this simulation process of the clocks of N cannot be extended to the ages of the stack symbols in the stack, since N' has limited access to its stack (it can only access the top part). To overcome this problem, we add to each stack symbol $\gamma \in \Gamma$ of N, its initial age,

and the time that has elapsed between the pushing of γ and the last time it was the topmost symbol. As in the case of the clocks of N, we can assume that the initial age and the time elapsed associated with each stack symbol of N' is in $[0\,..\,max] \cup \{\omega\}$. Now, every performed transition in N' should preserve that the age of the topmost stack symbol is given by the sum of its initial age and the time elapsed so far. Hence, when a symbol is popped from the stack, the elapsed time of the new topmost stack symbol must be updated by adding to it the elapsed time of the popped symbol. Formally, the pushdown system N' is defined as follows:

- The set Q' of states of N' is $Q \times [X \rightarrow ([0\,..\,max] \cup \{\omega\})]$. A state of N' is then of the form (q, ν) where $q \in Q$ is the current state of N and ν is a mapping from X to $[0\,..\,max] \cup \{\omega\}$. If a clock $x \in X$ has a value strictly greater than max in N then $\nu(x) = \omega$ in N', otherwise the value of the clock x in N is $\nu(x)$.
- The stack alphabet Γ' is $(\Gamma \times ([0\,..\,max] \cup \{\omega\})^2) \cup \{(\bot, 0, 0)\}$ where $(\bot, 0, 0)$ is a special symbol used to mark the bottom of the stack. A stack symbol of the form (γ, y, z) on the top of the stack of N' corresponds to the fact that γ is the topmost stack symbol of N such that if its initial age i is strictly greater than max then $y = \omega$, otherwise $y = i$. Moreover, if e is the elapsed time while γ is in the stack of N, then if e is strictly greater than max then $z = \omega$, otherwise $z = e$. Notice that the age of γ in N is $(i + e)$.
- The set Δ' is the smallest set of rules satisfying the following conditions:

  1. Simulating a discrete transition of N: Let $(\gamma, y, z) \in \Gamma'$ be a stack symbol of N'. Then, for every discrete transition $\delta \in \Delta$ of N of the form $(q, (\alpha_1, J_1) \cdots (\alpha_m, J_m)) \xrightarrow{(\phi, R)} (q', (\gamma_1, I_1) \cdots (\gamma_n, I_n))$, N' has a transition of the form: $((q, \nu), (\alpha_1, y_1, z_1) \cdots (\alpha_m, y_m, z_m)(\gamma, y, z)) \rightarrow ((q', \nu'), (\gamma_1, y'_1, z'_1) \cdots (\gamma_n, y'_n, z'_n)\ (\gamma, y', z'))$ if the following conditions hold:

  - For every clock $x \in X$, $\nu(x) \in \phi(x)$. The valuation of each clock x should satisfy the time constraint given by ϕ.
  - For every clock $x \in X$, $\nu'(x) = 0$ if $x \in R$, otherwise $\nu'(x) = \nu(x)$. Since no unit of time has been elapsed, only the clocks that are in R are reset.
  - For every $i : 1 \leq i \leq m$, we have $(y_i + \sum_{j=1}^{i} z_j) \in J_i$. This means that the age of the stack symbol α_i is given by the sum of the time elapsed at each stack symbol α_j with $j : 1 \leq j \leq i$ and its initial age y_i when it was pushed.
  - If $(\gamma, y, z) = (\bot, 0, 0)$ then $(\gamma, y', z') = (\bot, 0, 0)$. This means that the bottom stack symbol can never be popped.
  - If $\gamma \in \Gamma$ then let $e = z + \sum_{j=1}^{m} z_j$. If $e \leq max$ then $z' = e$, otherwise $z' = \omega$. The elapsed time of the new topmost stack symbol γ must be updated using the elapsed time of each individual symbol α_i with $j : 1 \leq j \leq m$. Moreover, we have $y' = y$ since the initial age of γ remains the same.

- For every $i : 1 \leq i \leq n$, we have $y'_i \in ([0\,..\,max] \cup \{\omega\}) \cap I_i$ and $z'_i = 0$. The newly pushed stack symbol γ_i has an initial age in I_i. Moreover, the elapsed time of γ_i is 0 since it is newly pushed into the stack.

2. Simulating a timed transition of N: For every state $q \in Q$, $\nu, \nu' \in [X \to ([0\,..\,max] \cup \{\omega\})]$, and $(\gamma, y, z), (\gamma, y', z') \in \Gamma'$, the pushdown system N' has a transition δ_{time} of the form $((q, \nu), (\gamma, y, z)) \to ((q, \nu'), (\gamma, y', z'))$ if the following conditions hold:

- For every clock $x \in X$, if $\nu(x)+1 \leq max$ then $\nu'(x) = \nu(x)+1$, otherwise $\nu'(x) = \omega$. Since a timed transition is performed, the pushdown system N' should update the valuation of all its clocks accordingly.
- If $(\gamma, y, z) = (\bot, 0, 0)$ then $(\gamma, y', z') = (\bot, 0, 0)$. This means that the bottom stack symbol can never be popped or modified.
- If $\gamma \in \Gamma$ then let $e = z + 1$. If $e \leq max$ then $z' = e$, otherwise $z' = \omega$. Moreover, we have $y' = y$. The elapsed time of the topmost stack symbol must be updated since one time unit has passed.

Finally, the relation between N and N' is given by the following lemma:

Lemma 7. $(q_0, \nu, \epsilon) \to_N^* (q_f, \nu, \epsilon)$ *iff* $((q_0, \nu), (\bot, 0, 0)) \to_{N'}^* ((q_f, \nu), (\bot, 0, 0))$.

Since the reachability problem for pushdown systems is decidable (see for example [15, 9]), and from Theorem 3 and Theorem 6, we can conclude that the cost-threshold problem is also decidable.

Corollary 8. *The cost-threshold problem for PTPS is decidable.*

Moreover, from Theorem 2 and Corollary 8, we obtain:

Corollary 9. *Let N be a priced timed pushdown system with an initial configuration $c_0 \in Conf(N)$ and a regular set F of final configurations. Then, it possible to compute the minimal possible costs of reaching F.*

6 Conclusion

We introduced the model of (discrete) timed pushdown systems which is an extension of pushdown systems. We showed that the reachability problem for timed pushdown systems is decidable and can be reduced to the reachability problem for standard pushdown systems (which is a decidable problem). Moreover, we have considered priced timed pushdown systems which is an extension of timed pushdown systems with a cost model. We proved that the cost-threshold problem is decidable (by a reduction to the reachability problem for timed pushdown systems). As a consequence of this result, the minimal cost reachability problem for priced timed pushdown systems is decidable.

A challenging problem which we are currently considering is to extend our results to the case of dense-timed pushdown systems.

References

1. P. A. Abdulla and R. Mayr. Minimal cost reachability/coverability in priced timed petri nets. In *FOSSACS*, LNCS 5504, pages 348–363. Springer, 2009.
2. P. A. Abdulla and R. Mayr. Computing optimal coverability costs in priced timed petri nets. In *LICS*, 2011.
3. R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *ICALP*, LNCS 3142, pages 122–133. Springer, 2004.
4. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
5. R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. *Theor. Comput. Sci.*, 318(3):297–322, 2004.
6. G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. W. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC*, LNCS 2034, pages 147–161. Springer, 2001.
7. M. Benerecetti, S. Minopoli, and A. Peron. Analysis of timed recursive state machines. In *TIME*, pages 61–68. IEEE Computer Society, 2010.
8. A. Bouajjani, R. Echahed, and R. Robbana. On the automatic verification of systems with continuous variables and unbounded discrete data structures. In *Hybrid Systems*, LNCS 999, pages 64–85. Springer, 1994.
9. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR*, LNCS 1243, pages 135–150. Springer, 1997.
10. P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS*, LNCS 3328, pages 148–160. Springer, 2004.
11. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. Binary reachability analysis of discrete pushdown timed automata. In *CAV*, LNCS 1855, pages 69–84. Springer, 2000.
12. M. Emmi and R. Majumdar. Decision problems for the verification of real-time software. In *HSCC*, LNCS 3927, pages 200–211. Springer, 2006.
13. J. Esparza and J. Knoop. An automata-theoretic approach to interprocedural data-flow analysis. In *FoSSaCS*, LNCS 1578, pages 14–30. Springer, 1999.
14. G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952.
15. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
16. T.W. Reps, S. Schwoon, and S. Jha. Weighted pushdown systems and their application to interprocedural dataflow analysis. In *SAS*, LNCS 2694, pages 189–213. Springer, 2003.
17. S. Schwoon. *Model-Checking Pushdown Systems*. PhD thesis, Technische Universität München, 2002.
18. A. Trivedi and D. Wojtczak. Recursive timed automata. In *Proceedings of the 8th international conference on Automated technology for verification and analysis*, ATVA, pages 306–324, 2010.
19. R. Valk and M. Jantzen. The residue of vector sets with applications to decidability problems in petri nets. *Acta Inf.*, 21:643–674, 1985.